OPEN ACCESS
J. OF ICT, VOL. 3, NO. 1, PP.065-078, MAY 2019
ISSN: 2686-1089 (ONLINE)

Journal of Informatics and Communications Technology (JICT)

Automated Functional Testing pada API menggunakan Keyword Driven Framework

Syifa Nurgaida Yutia

Teknologi Informasi, Institut Teknologi Telkom Jakarta
Jalan Daan Mogot KM.11, Jakarta 11710 Indonesia
syifanurgaida@ittelkom-jkt.ac.id

Abstract

Pada era digital saat ini, penggunaan API (*Application Programming Interface)* sangat penting dalam proses pengembangan aplikasi. API berperan dalam menjembatani antara dua fungsi atau dua perangkat lunak yang berbeda untuk pertukaran data. Pengujian secara manual masih memiliki masalah antara lain adalah *not reusable* dan tidak efisien dengan perubahan. Sedangkan, *automation testing* menggunakan skrip uji yang berisi objek atau elemen pada perangkat lunak yang direkam untuk dapat dijalankan pada pengujian secara berulang dan dapat disesuaikan jika terdapat perubahan. Pengujian *functional* untuk memastikan fungsi API berjalan dengan baik dapat lebih mudah dilakukan dengan *automation testing*. Salah satu metode pada *automation testing* yaitu *keyword driven framework* dengan menggunakan *keyword* untuk mendeskripsikan langkah-langkah pengujian. Kelebihan yang ditawarkan *automated functional testing* menggunakan *keyword driven framework* yaitu *test case* ringkas, mudah dimengerti dan mudah dimodifikasi. Makalah ini mengusulkan *automated functional testing* pada API menggunakan *keyword driven framework* dengan penerapannya menggunakan tools *automation testing* yaitu *Robot Framework*.

Keywords: *automation testing, functional testing*, *API, keyword driven framework, keyword driven testing, robot framework.*

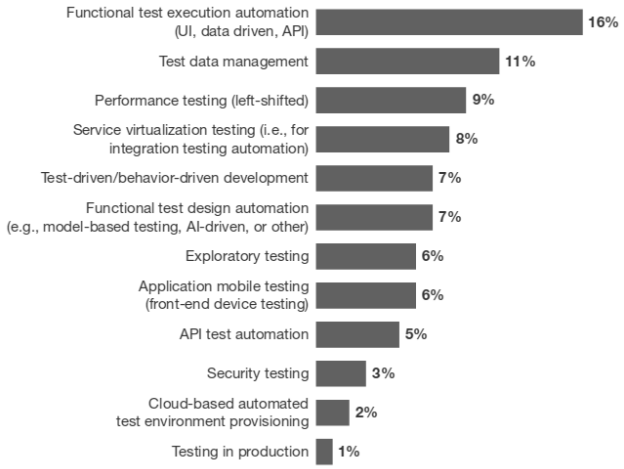
I. PENDAHULUAN

Perkembangan pembuatan perangkat lunak saat ini tidak terlepas dari peran API (Application Programming Interface). Penggunaan API meningkat dengan semakin berkembangnya teknologi IoT dan Cloud Computing untuk membuat antar device saling terhubung dan bertukar data. API berperan sebagai jembatan dalam komunikasi antar perangkat lunak yang menghubungkan dengan perangkat lunak pihak ketiga tanpa intervensi dari manusia. API terdiri dari object-object dan method-method yang terletak di suatu server yang terhubung ke internet sehingga dapat diakses menggunakan protokol HTTP dan SOAP (Simple Object Access Protocol). Pengujian API sangat penting dilakukan karena untuk menjamin apakah sistem dapat berfungsi dengan baik. Pengujian API meliputi pengujian fungsional dan pengujian kinerja [1].

Pengujian merupakan proses penting dalam siklus pengembangan perangkat lunak untuk mendeteksi perbedaan antara kondisi yang ada dengan kondisi yang diinginkan defect/errors/bugs dan mengevaluasi fitur-fitur dari entitas perangkat lunak [2]. Tim pengembang juga menguji sistem untuk menghasilkan produk berkualitas tinggi dan memenuhi requirement dari klien dengan lebih baik [3]. Namun, ketika sebuah perangkat lunak menjadi lebih rumit dan perlu dikembangkan dalam proses pengembangan yang lebih cepat. Maka untuk meningkatkan efisiensi pengujian, cakupan pengujian, dan keefektifan pengujian, organisasi mulai menggunakan automation testing.

Automation testing berbeda dengan pengujian manual karena proses pengujian yang dilakukan dengan menyusun scripted test. Kelebihan automation testing dibandingkan pengujian manual yaitu dapat dijalankan berulang kali dengan biaya yang lebih rendah dan kecepatan lebih cepat, dan saat ini menjadi sangat penting dalam proses pengembangan perangkat lunak [3]. Berdasarkan hasil survey pada Gambar 1 yang dilakukan

oleh lembaga riset Forrester pada tahun 2019, layanan paling berdampak yang diberikan oleh vendor dalam menghasilkan perangkat lunak yang lebih baik dengan lebih cepat yaitu Functional Test Execution Automation yang meliputi (UI, Data Driven, API) dengan memperoleh 16% [4].



Gambar 1. The Most impactful Continuous Testing services [4]

Pengujian functional dilakukan untuk memeriksa dan memastikan fungsi-fungsi yang terdapat pada perangkat lunak yang sedang dibangun berfungsi sesuai dengan yang diharapkan. Salah satu metode pada automation testing yaitu keyword driven framework. Metode keyword driven framework merupakan pengujian yang berbasis kata kunci atau keyword dengan cara menggambarkan kasus uji menggunakan seperangkat kata kunci yang telah ditentukan. Kata kunci ini adalah nama yang dikaitkan dengan serangkaian tindakan yang diperlukan untuk melakukan langkah spesifik dalam kasus uji. Dengan menggunakan kata kunci untuk mendeskripsikan langkah-langkah pengujian maka uji kasus dapat lebih mudah dipahami, dipertahankan, dan diotomatisasi [5]. Dalam makalah ini mengusulkan automated functional testing pada API dengan menggunakan keyword driven framework dan tools automation testing yaitu Robot Framework. Penerapan keyword driven framework pada pengujian memudahkan pengguna yang tidak memiliki pengetahuan pemrograman untuk membuat script dan menjalankannya secara otomatis, menguji aplikasi dan menghasilkan hasilnya. Bagian makalah ini disusun sebagai berikut: Bagian II menyajikan Landasan Teori. Pada bagian III Metodologi Penelitian, bagian IV Implementasi dan akhirnya pada bagian V yaitu berisi Kesimpulan.

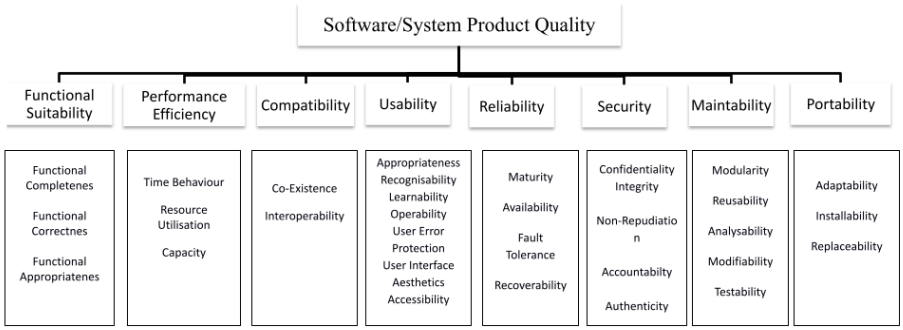
II. LANDASAN TEORI

A. *Automation Testing*

Automation Testing merupakan pelaksanaan kasus uji dengan cara otomatis tanpa intervensi manual dengan hanya merekam dan memutar kembali yang membuat *tester/*penguji dapat mengulangi pekerjaan di setiap siklus pengujian [6]. Dengan adanya *automation testing* dapat digunakan untuk mengurangi upaya pengujian dan mencapai pelaksanaan pengujian yang efisien. Hal ini dapat diterapkan dengan menggunakan *tools automation testing* yang sedang marak digunakan diantaranya yaitu *robot framework*, katalon studio, selenium.

B. *Functional Testing*

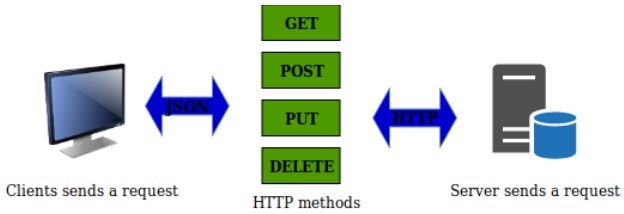
Functional Test merupakan jenis pengujian yang bertujuan untuk menentukan apakah persyaratan fungsional dari item tes telah dipenuhi atau contoh, termasuk dapat mengidentifikasi apakah suatu fungsi telah dilaksanakan sesuai dengan persyaratan yang ditentukan. itu dapat dilakukan dengan menggunakan teknik desain uji berbasis spesifikasi dan struktur [7].



Gambar 2. ISO/IEC 25010 Product Quality Model

C. API

API (*Application Programming Interface)* merupakan kumpulan dari berbagai elemen seperti *function, protocols, dan tools* yang digunakan para pengembang perangkat lunak untuk mengembangkan perangkat lunak mereka [8]. API memudahkan bagaimana komponen perangkat lunak dapat berinteraksi. Salah satu implementasi dari API adalah REST API. REST (*Representational State Transfer*) adalah suatu arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data dan metode ini sering diterapkan dalam pengembangan aplikasi. Dimana tujuannya adalah untuk menjadikan sistem yang memiliki performa yang baik, cepat dan mudah untuk di kembangkan (*scale*) terutama dalam pertukaran dan komunikasi data. REST API dapat digunakan oleh situs atau aplikasi apa pun, tidak peduli bahasa apa yang ditulisnya karena permintaan didasarkan pada protokol HTTP universal, dan informasi yang dikembalikan dalam format JSON yang dapat dibaca oleh hampir semua bahasa pemrograman seperti pada Gambar 3.



Gambar 3. Cara kerja REST API

Method yang digunakan pada REST API yaitu GET, POST, PUT dan DELETE. Untuk melihat satu data atau daftar dapat menggunakan metode GET. Method untuk membuat atau mengirim data dapat menggunakan metode POST. Metode PUT dapat digunakan untuk mengedit data, dan metode DELETE untuk menghapus.

D. *Keyword-Driven Testing*

Keyword-Driven Testing adalah pendekatan spesifikasi *test case* (kasus uji) yang biasanya digunakan untuk membangun *automation testing* dan pengembangan *framework automation testing*. *Keyword-Driven Testing* berdasarkan [9] adalah cara menggambarkan *test case* dengan menggunakan serangkaian *keyword* (kata kunci) yang telah ditentukan. *Keyword* dalam pengujian ini bisa disebut *function* atau dalam istilah pemrograman yaitu fungsi untuk melakukan tindakan. *Keyword* juga bisa disebut kombinasi dari satu atau lebih aksi pada suatu *tes object*. Dengan menggunakan *keyword* untuk mendeskripsikan langkah-langkah pengujian, sehingga pengujian dapat lebih mudah dipahami, dipelihara, dan diotomatisasi. Manfaat pengujian berdasarkan *keyword* meliputi kemudahan penggunaan, mudah dipahami bagi yang belum memiliki *skill* pemrograman dan mudah dilakukan *maintenance*.

Tujuan utama pengujian menggunakan *keyword driven testing* adalah untuk menyediakan serangkaian *keyword* dasar yang cukup komprehensif sehingga sebagian besar *test case* yang ada dapat mencakup seluruhnya dari *keyword* ini. Untuk *automation testing*, setiap *keyword* perlu diimplementasikan dalam perangkat lunak. Contoh penerapan *keyword driven testing* yaitu pada Tabel 1.

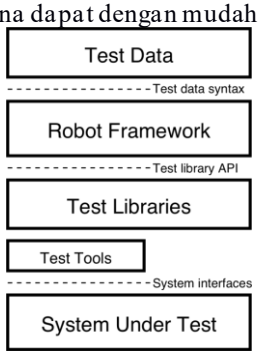
TABLE I
CONTOH PENERAPAN KEYWORD DRIVEN TESTING

Keyword	Object	Value
Open Browser	Chrome	www.google.com
Set Text	Search TextBox	"apa itu keyword driven testing?"
Press Button	Search Button	

Dalam pengujian yang digerakkan oleh *keyword,* setiap tindakan pengujian seperti yang ditunjukkan pada Tabel 1 terdapat *keyword* seperti *open browser, set text* dan *press button* karena setiap tindakan dalam skenario pengujian seperti penekanan tombol, membuka atau menutup *browser* dideskripsikan menggunakan *keyword*.

E. *Robot Framework*

Robot Framework adalah *keyword-driven test automation* untuk pengujian penerimaan *end to end* dan *Acceptance-Test-Driven Development* (ATDD) [11]. *Keyword library* ditulis dalam bahasa Python atau Java. *Framework* ini gratis dan dapat digunakan untuk melakukan pengujian sistem, *functional test* dan *regression testing*. Perangkat lunak ini mendukung struktur tes tingkat tinggi dan menyediakan beberapa editor tes seperti RIDE atau plugin Eclipse sehingga pengguna dapat dengan mudah memelihara dan menskalakan tes [10].

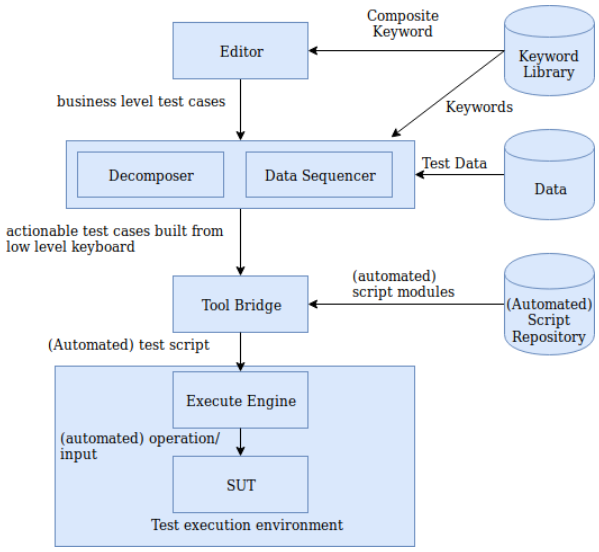


Gambar 5. *Robot Framework Architecture* [10]

Gambar 5 merupakan arsitektur dari *Robot Framework,* ketika eksekusi tes dimulai, pertama yang dilakukan yaitu mem-parsing data uji. Kemudian menggunakan *keyword* yang disediakan oleh *tes library* untuk berinteraksi dengan sistem yang diuji. Interaksi antara *test libraries* dan sistem yang diuji biasanya terjadi langsung. Namun, beberapa *test libraries* harus memiliki driver seperti Selenium2Library, atau REST (untuk pengujian API) agar dapat terhubung dengan sistem yang diuji. Robot Framework dapat bekerja dengan beberapa browser seperti Internet Explorer, Firefox dan Chrome. Sebagai hasilnya mendapatkan laporan dalam format HTML serta output XML.

III.METODOLOGI PENELITIAN

Pada penelitian ini komponen fungsional yang membentuk *Framework* pada *Keyword Driven Automation Testing* digambarkan pada Gambar 4 berikut berdasarkan [9].



Gambar 4. *Keyword Driven Testing Framework* [9]

Deskripsi komponen pada *Keyword Driven Testing Framework* adalah sebagai berikut, *editor* digunakan untuk menyusun *test case* dengan menggunakan *keyword*. Sumber *keyword* berasal dari *keyword library*. Dekomposer (Pengurai) diperlukan jika *keyword* komposit digunakan. Tugas utama dekomposer adalah untuk mengubah *test case* yang terdiri dari urutan *keyword* tingkat tinggi, ke dalam urutan yang sesuai *keyword* tingkat rendah. *Sequencer data* diperlukan jika pengujian berbasis *keyword* harus diterapkan dengan beberapa set data yang terkait dalam suatu *test case*. Tugas utama dari *sequencer data* adalah mengubah urutan *keyword* (mis. level rendah atau level tinggi) yang belum terkait dengan dari daftar *keyword* ke data tertentu. Dengan melakukan ini, daftar tindakan dapat ditulis hanya sekali di *editor* yang digerakkan oleh *keyword* sehingga dapat diulang untuk set data yang diinginkan. Semua parameter diganti dengan nilai akhir yang dibutuhkan pada masing-masing *test case*.

Selanjutnya, *tool bridge* hanya diperlukan untuk pelaksanaan tes otomatis. Tugas *tool bridge* adalah untuk menyediakan koneksi antara *keyword*, seperti yang muncul dalam *keyword test case* atau di *keyword library*, dan implementasi terkait di *test execution environment*. Untuk setiap *keyword* yang dilewatkan dari sekuens data atau dari dekomposer, *tool bridge* akan tergantung pada implementasinya, meminta *test execution engine* untuk memanggil skrip yang tepat (mis. Kode eksekusi *keyword*), berfungsi, dan melakukan tindakan yang benar dengan data yang sesuai. Dalam praktiknya, *tool bridge* dapat diimplementasikan sebagai perangkat lunak terpisah, sebagai skrip dalam *automation tool*, atau sebagai bagian dari alat manajemen pengujian. Beberapa implementasi *tool bridge* mungkin disebut sebagai "*generator*," misalnya ketika skrip atau bagian skrip dihasilkan untuk dieksekusi oleh *automation tool*.

Selain itu, *tool bridge* dapat disebut "juru bahasa" atau "mesin", misalnya, ketika skrip dijalankan untuk menafsirkan urutan kata kunci dan memanggil sub-fungsi yang sesuai. *Script repository* menyimpan kode eksekusi *keyword*. Ini hanya diperlukan jika Pengujian Berbasis Kata Kunci dilakukan dengan tujuan mengeksekusi *test case* secara otomatis. Untuk otomatisasi Pengujian Berbasis *keyword*, setiap *keyword* perlu dikaitkan dengan setidaknya satu perintah, skrip atau fungsi pengujian, yang mengimplementasikan tindakan yang terkait dengan *keyword* tersebut. Dalam praktiknya, *script repository* sering diimplementasikan oleh alat otomatisasi pengujian atau disimpan di sebuah lokasi yang ditentukan dalam sistem file. *Execution Engine* adalah alat yang diimplementasikan baik oleh perangkat lunak, perangkat keras atau keduanya. Tugasnya adalah untuk menjalankan *test case* dengan melakukan tindakan yang terkait dengan *keyword*. Penerapan *Execution Engine* bervariasi tergantung pada objek pengujian dan *environment*. *Execution Engine* dapat menjadi alat eksekusi uji komersial, (misal Alat tangkap dan pemutaran, atau bisa juga berupa alat perangkat keras, yang dikendalikan oleh perangkat lunak, seperti robot).

IV. IMPLEMENTASI

Untuk penerapan pada penelitian ini dengan menggunakan tools Robot Framework dalam pengujian functional API.

A. Perancangan API

Pada penelitian ini API yang digunakan adalah sampel fake online REST API yang sudah tersedia untuk pengujian dari JSONPlaceholder dan hanya menggunakan dua method yaitu GET dan POST sebagai berikut yang dijelaskan pada Tabel 1.

TABLE 1
Metode yang digunakan pada pengujian

REST: https://jsonplaceholder.typicode.com/		
Method	**URL**	**OUTPUT**
GET	/users/1	Data user dengan id tertentu (contoh id=1)
	/users	Data semua user
	/users?_limit=5	Data user dengan batas limit 5
POST	/posts/users	Menambah data user

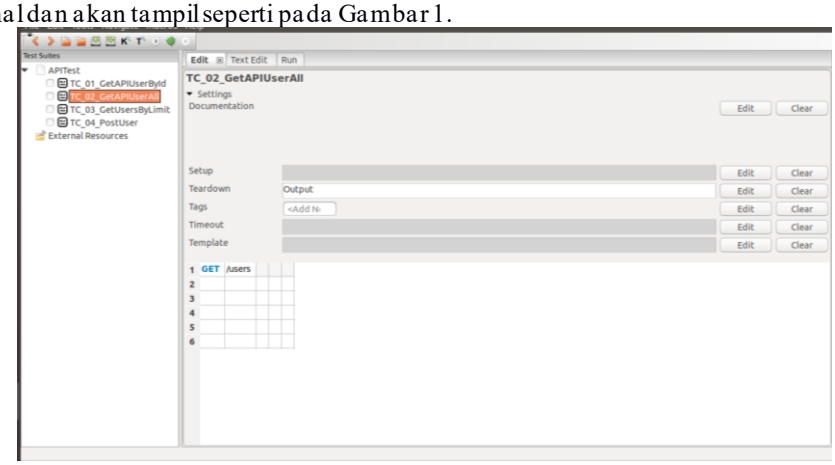
B. Instalasi Robot Framework

Pada penelitian ini instalasi Robot Framework dilakukan pada operating system linux Ubuntu 18.04, dan penerapan testing menggunakan editor RIDE (Robot Framework IDE). Pada instalasi robot framework terdapat langkah-langkah yang harus dilakukan yaitu:

1. Install Python
   Syarat minimum python yang terinstall yaitu versi python 2.6. Untuk mengecek apakah sudah terinstall dengan perintah python –-version. Jika tidak tersedia, maka bisa menginstal python dengan perintah apt-get install python.
2. Install pip
   Pip merupakan penginstal paket Python yang digunakan untuk menginstal apapun yang berhubungan dengan Python. Instalasi pip dengan mengetik perintah pada terminal sudo apt-get install python-pip.
3. Install wxPython
   RIDE (Robot Framework IDE) dapat diimplementasikan menggunakan wxPython toolkit. Instalasi wxPython dengan perintah sudo apt-get install python-wxgtk2.8.
4. Install Robot Framework
   sudo pip install robotframework
5. Install RIDE (Robot Framework IDE)
   RIDE merupakan editor dari robot framework yang akan digunakan dalam penelitian ini, instalasi RIDE dengan menggunakan perintah pip install robotframework-ride atau dapat menggunakan easy_install robotframework-ride.
6. Install Selenium Library
   Selenium Library merupakan library eksternal yang dapat digunakan dengan perintah install sudo pip install robotframework-seleniumlibrary.
7. Install RESTinstance

RESTinstance merupakan library eksternal untuk pengujian (RESTful) JSON APIs yang dapat diinstall dengan perintah pip install --upgrade RESTinstance.

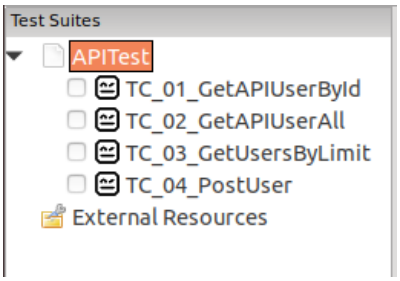
Setelah step-step dalam proses instalasi selesai dilakukan, maka RIDE dapat digunakan dengan perintah ride.py pada terminal dan akan tampil seperti pada Gambar 1.



Gambar 1 Antarmuka RIDE

C. *Test Suite*

Test Suite merupakan kumpulan atau wadah yang memiliki serangkaian tes yang membantu penguji dalam melaksanakan dan melaporkan status pelaksanaan tes. *Test Suite* terdiri dari beberapa *Test Case*. Sedangkan *Test Case* merupakan serangkaian tes yang digunakan untuk menentukan apakah rangkaian pengujian bekerja dengan baik.



Gambar 2 Test Suite dan Test Case yang dibuat pada RIDE

Test Suite yang dibuat bernama APITest dan memiliki 4 buah Test Case seperti yang tertera pada gambar 2. *Code test case* yang dibuat didalam test suite adalah sebagai berikut

```
*** Settings ***
Library           SeleniumLibrary
Library           REST    https://jsonplaceholder.typicode.com

*** Test Cases ***
TC_01_GetAPIUserById
    GET    /users/1
    Integer    response body id    1
    [Teardown]    Output

TC_02_GetAPIUserAll
    GET    /users
    [Teardown]    Output

TC_03_GetUsersByLimit
    GET    /users?_limit=5
    [Teardown]    Output    $[*].id    # all ids as a json array

TC_04_PostUser
    POST    /users    { "id": 11,"name": "syifa nurgaida","username": "Syifa","email": "syifayutia@gmail.com","address": {"s
    Integer    response status    201
```

Gambar 3 *Code* pada *Test Suite*

Pada gambar 3 terlihat bahwa *test suite* menggunakan library SeleniumLibrary dan RESTinstance, dan ke-4 test case yang tertera yaitu TC_01_GetAPIUserById untuk mengambil data user yang memiliki id 1 dengan perintah GET /users/1. Pada TC_02_GetAPIUserAll yaitu untuk mengambil semua data user dengan menggunakan perintah GET /users, sedangkan untuk TC_03_GetUsersByLimit yaitu untuk mengambil data user dengan limit atau batas 5 dengan perintah GET /users?_limit=5. Yang terakhir yaitu TC_04_Post User digunakan untuk mengirimkan/membuat data dengan perintah

POST /users { "id": 11,"name": "syifa nurgaida","username": "Syifa","email": "syifayutia@gmail.com","address": {"street": "Cimandiri","suite": "cipayung","city": "Tangerang Selatan","zipcode": "15411","geo": {"lat": "-37.3159","lng": "81.1496"}},"phone": "0828739828","website": "syifatumblr.com","company": {"name": "IDX","catchPhrase": "family oriented", "bs": "emarkets"}}

D. *Test Case Execution and Report*

Terdapat 4 *test case* yang akan di eksekusi yang pertama yaitu TC_01_GetAPIUserById yang akan menampilkan user berdasarkan id.

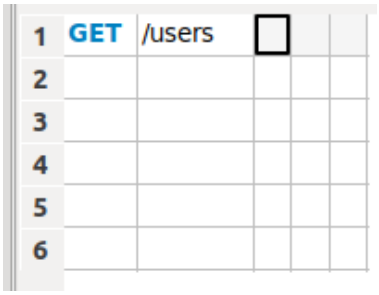
1	GET	/users/1			
2	Integer	response body id	1		
3					
4					
5					
6					
7					

Gambar 4 GET request untuk user yang spesifik

Kemudian klik tombol *run test* untuk menghubungi server dan mengeluarkan permintaan GET untuk informasi tentang pengguna dengan ID 1, dan dengan kata kunci INTEGER untuk memvalidasi data yang akan didapatkan yaitu data pengguna dengan ID 1 yang ada di respon body. Pada Gambar 5 menunjukan bahwa pengujian berhasil dan output yang didapatkan dari server adalah data user dengan id 1 dalam array JSON.

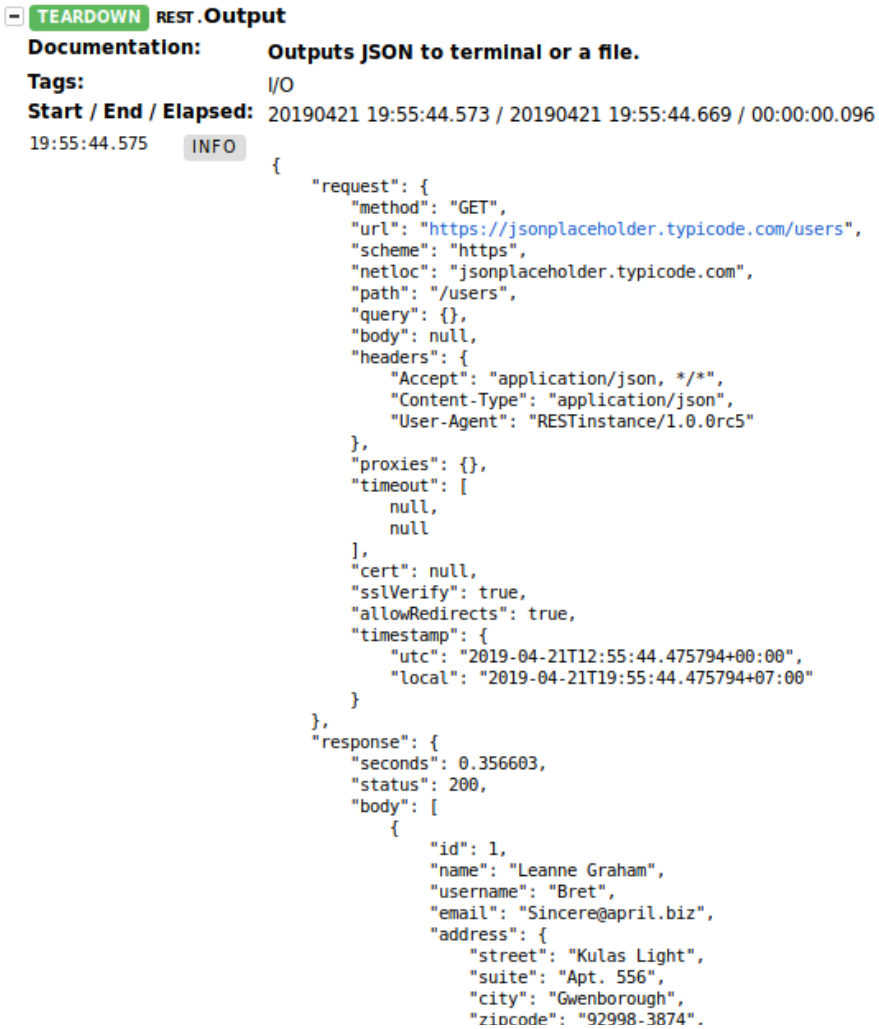
Gambar 5 *Report* TC_01_GetAPIUserById

Test Case selanjutnya yaitu TC_02_GetAPIUserAll yang akan menampilkan seluruh data user.



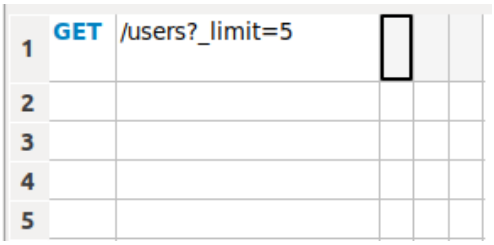
Gambar 6 GET request untuk seluruh data user

Results dan *report* sebagai berikut:



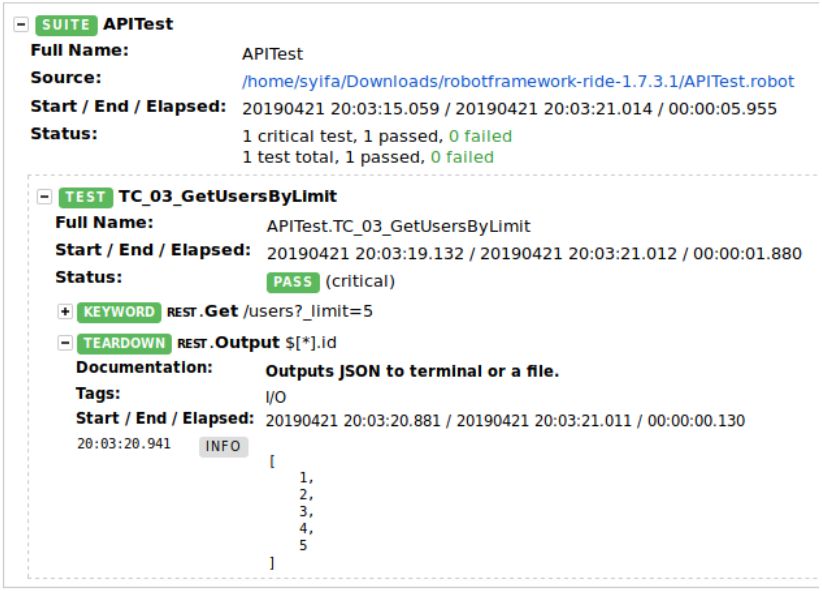
Gambar 7 *Report* TC_01_GetAPIUserAll

Seperti yang terlihat pada Gambar 7 bahwa pengujian berhasil dan output yang didapatkan dari server adalah seluruh data user dalam array JSON. Selanjutnya yaitu TC_03_ GetUsersByLimit yang akan menampilkan data user dengan limit/batas tertentu



Gambar 8 GET request data user berdasarkan limit

Dengan argumen / users? _Limit = 5 ke *keyword* GET maka data yang ditampilkan dibatasi dengan jumlah user hanya 5 yang diambil informasi dari server, dan dengan argumen tersebut juga memerintahkan framework untuk mendapatkan hanya ID dari pengguna dan masukkan mereka dalam array JSON sehingga *results* dan *report* sebagai berikut:



Gambar 9 *Report* TC_03_GetUsersByLimit

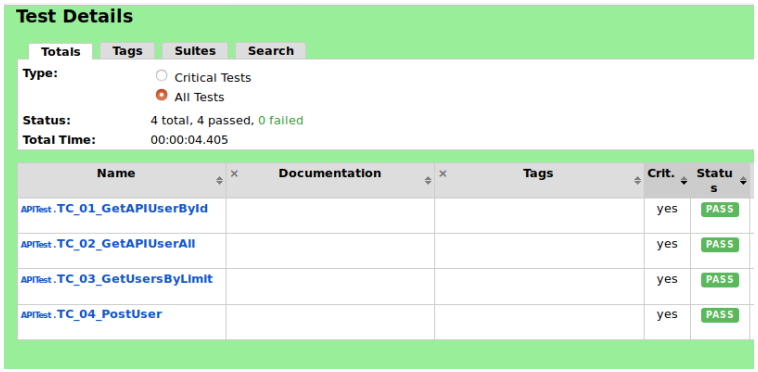
Seperti yang terlihat pada Gambar 9 bahwa pengujian berhasil dan output yang didapatkan dari server adalah ID pengguna yang diurutkan dalam array JSON. Kemudian *Test Case* yang terakhir yaitu TC_04_PostUser yang digunakan untuk mengirimkan/menambah data user

Gambar 10 POST untuk data user

Pada sistem yang diuji sebelumnya terdapat data user dengan total 10 pengguna, sehingga pada perintah POST data yang akan ditambah yaitu id 11 dan dengan menambahkan verifikasi *keyword* INTEGER dengan status respons 201 yang berarti berhasil. Kemudian dieksekusi dan menghasilkan *report sebagai berikut*

Gambar 11 *Report* TC_04_PostUser

Dari ke 4 *test case* yang sudah diuji semua berhasil dan menghasilkan data yang diharapkan dengan rangkuman yang terlihat pada Gambar 12.



Gambar 12 *Report* Status Pengujian Seluruh *Test Case*

V. Kesimpulan

Pengujian dalam proses pengembangan perangkat lunak sangat berpengaruh untuk menghasilkan produk/software yang berkualitas. Dalam menentukan jenis pengujian yang dipilih harus sesuai dengan kualitas atau output yang diinginkan, seperti pengujian fungsional dilakukan untuk memastikan kualitas software sesuai dengan fungsi yang diharapkan dan berjalan baik. Penggunaan automation testing dapat mempermudah proses pengujian. Kelebihan automation testing dibanding pengujian manual yaitu dapat meningkatkan efektivitas, efisiensi, dan coverage dalam testing. Automation testing bisa menjadi solusi untuk menghemat waktu dengan mengotomatiskan proses testing yang sifatnya berulang atau repetitive dan untuk memastikan fungsionalitas yang penting dan critical berjalan dengan baik. Penerapan automation testing juga harus mempertimbangkan beberapa hal yakni tentukan kelayakan produk pengujian otomatis, buat penilaian risiko untuk pengujian proyek perangkat lunak, kembangkan rencana pengujian yang cermat, kombinasi dua program: pengujian manual dan pengujian otomatis dan meringkas proses pengujian otomatis perangkat lunak. Pengujian automation testing untuk API dengan berdasarkan keyword memudahkan seorang QA (Quality Assurance) dalam membuat test case (kasus pengujian) dengan menentukan keyword. Untuk perancangan automation testing yakni membuat test suite, desain test case , test script, eksekusi uji, evaluasi tes dengan melihat hasil report. Penerapan Tools Robot Framework dalam pengujian API telah berhasil dalam menghasilkan report pengujian dan test case yang dieksekusi berjalan lancar.

REFERENCES

[1] X.Y. Guo, X.S. Qiu, Y.H. Chen, and F. Tang, "Design and Implementation of Performance Testing Model for Web Services," *International Conference on Artificial Intelligence and Computational Intelligence,* 2009.

[2] IEEE Standard 1059-1993, "IEEE Guide for Software Verification and Validation Plans", pp.1-92, IEEE Computer Society, 1993.

[3] Y. Wang, "Test Automation Maturity Assessment," IEEE 11th *International Conference on Software Testing, Verification and Validation (ICST),* 2018.

[4] D.L. Giudice, C. Mines, A.Dobak, and A. reese, "The Forrester Wave™: Global Continuous Testing Service Providers, Q1 2019," Forrester, 2019.

[5] S. Eldh, K. Andersson, A. Ermedahl, K. Wiklund, "Towards a test automation improvement model (TAIM)", *Software Testing Verification and Validation Workshops (ICSTW) 2014 IEEE Seventh International Conference on*, pp. 337-342, 2014.

[6] L. Feng, S. Zhuang, "Action-Driven Automation Test Framework for Graphical User Interface (GUI) Software Testing', IEEE Autotestcon, pp.22-27, 2007.

[7] ISO 29119-4, "Software and systems engineering –Software testing- Part 4: Test Techniques. Standard, International Organization for Standardization," Geneva, CH, 2015.

[8] F. Thung, "API Recommendation System for Software Development", Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, pp.896-899, 2016.

[9] ISO 29119-5, "Software and systems engineering– Software Testing-Part 5: Keyword-Driven Testing. Standard, International Organization for Standardization", Geneva, CH, 2016.

[10] V. Garousi and F. Elberzhager, "Test Automation: Not Just for Test Execution", IEEE Software, pp.90 – 96, 2017.